



Data Design Corporation
Gaithersburg, MD

**TR122 CPCI MODULE
2 CHANNEL 100 MHz, 200 MS/S
TRANSIENT RECORDER**

April 2005

Data Design Corporation
7851-A Beechcraft Avenue
Gaithersburg, MD 20879
WWW.DATADESIGNCORP.NET
(301) 670-1157

CONTENTS

1.0 Introduction	3
1.1 Specifications Summary	4
1.2 Setup and Installation	5
1.3 Front Panel	6
2.0 Using TR122 Software	7
2.1 Customizing The Display	8
2.2 Oscilloscope Mode	9
2.2.1 Vertical Settings	9
2.2.2 Horizontal Settings	10
2.2.3 Trigger Settings	11
2.2.4 Saving Waveforms And Settings	13
2.3 Transient Recorder Mode	14
2.3.1 Configuring Transient Record Memory	15
2.3.2 Performing A Transient Recording	17
2.3.2 Saving A Transient Record	18
3.0 TR122 Software Source Code	20
3.1 TR122 Software Architecture	21
3.2 The TR122 API	23
3.2.1 Calibration Functions	25
3.2.2 Abstract Instrument Functions	26
3.2.3 Display Functions	28
3.2.4 The Auto Thread	30
4.0 Calibration	31
4.1 Using Calibration Features Of The TR122 Software	32
4.1.1 Analog Frequency Compensation Alignment	33
4.1.2 Stored Calibration Controls	34
4.1.3 Adjusting The Calibration Data	35

1.0 Introduction

The TR122 is a high performance two channel transient waveform digitizing instrument designed in a single slot CPCI module and capable of digitizing to 12 bits at rates up to 200 million samples per second (MS/s) on both channels. Each channel includes an independent analog front end and analog-to-digital converter providing superior interchannel isolation and full bandwidth availability on all channels. The TR122 segmented memory architecture and internal timer allow multiple sequential events to be captured and time stamped without the need to read data between events. A two million sample static memory is provided for each channel and is divisible into smaller data segments.

The TR122 analog inputs have nine 1 volt to 500 volt full scale ranges selectable in 1,2,5 steps. A full scale input offset adjustment is also available. The effective sample rate is adjustable from 1 MS/s to 200 MS/s in eight steps. Each TR122 unit is always an independent instrument, but can coexist in hardware and software with additional units and other instruments in the same CPCI system.

The TR122 can respond to trigger events from either channel, from an external trigger input, or from software. A trigger event stops the continuous recording of a circular buffer of samples when a specified number of samples after the trigger have been recorded. The number of samples recorded after the trigger is adjustable. This approach allows the user to select an appropriate number of pre trigger samples to be left in memory when the record is complete. A binary status recorded with the data indicates whether each sample was taken before or after the trigger event. The TR122 software automatically sorts the data to chronological order, making the intricate details of the records transparent to the user.

The sample memory can be segmented in multiples of 64K samples. A 32-bit (4 billion count) internal timer is used to record a count with each trigger event which can be used to determine the relative time of multiple trigger events with microsecond accuracy. These time stamps are saved in a first-in-first-out (FIFO) memory which can be accessed through the TR122 software to determine how many triggers events are present in the sample memory and when they occurred with respect to each other.

While the TR122 is not an oscilloscope, the software is designed with much of the look and feel of an oscilloscope for the benefit of instant user familiarity. The instrument and connected devices can be configured in an oscilloscope mode and then switched to transient recorder mode to take advantage of powerful transient recorder features.

Finally, the TR122 software source code is provided as open source. In a world of data acquisition now muddied by thousands of semi-standard and custom solutions it makes sense to provide a properly layered design accessible to the user for unique needs. The principles of the software design are defined in this manual and are intended to allow the advanced user to peel back the layers and snap in other front end software. Such efforts are always underway and popular additions will be published on the Data Design website as they become available.

1.1 Specifications Summary

Dataway Interface

Compact PCI at 32 Bits and 33 MHz
Compliant with PICMG 2.0 R3.0
Single Width 3U Compact PCI Card

Sample Rate

200 MS/s All Channels
100 MHz Bandwidth

Input Channel Architecture

2 Independent Analog Signal Inputs
1 External Trigger Input

Analog Input Characteristics

Single Ended DC Or AC Coupled
Full Scale Offset 10-Bit Resolution
1 M Ω 20 pF Or 50 Ω Impedance

External Trigger Input

Single Ended DC Coupled
1 K Ω 20 pF

Ambient Temperature Range

0 To 70 C

Memory

2 MSample Per Channel
64 KSample Minimum Block Size
Configurable Segmenting

Input Sensitivity

12-Bit Resolution
1-500 Volts Full Scale In 9 Ranges
0.05% Accuracy (10 LSB)
80 dB Minimum Interchannel Isolation

Input Withstand Rating

1 M Ω Input \pm 250 VDC
50 Ω Input 4 Watts (14 Volts RMS)

Trigger Input Sources

Channel Greater Than Full Scale Range
External \pm 10V 10-Bit Resolution
Rising Or Falling Edge

1.2 Setup and Installation

The TR122 is a single width 3U CPCI module. To insert the TR122 in the CPCI crate, choose any convenient slot and slide the module into the crate with its top and bottom mated to the guide rails. Be sure that the module is properly aligned with the connector at the back of the crate. Push the module toward the back of the crate with gentle pressure. When the card pushes back, push the locking lever down and press the card firmly into the crate with firm pressure on the top and bottom of the front panel. The locking lever will raise and lock into place. Fasten the retaining screws at the upper end of the front panel and at the bottom of the front panel below the lever. To remove the card, reverse the process by removing the retaining screws and pressing down firmly on the locking lever to disengage the card from the connector at the rear of the crate.

NOTE: The module should be inserted in the crate only after turning off the power to the crate. Otherwise, damage to the module or system is possible due to momentary misalignment of pins on the connector. CPCI hot swap operation is not supported.

It is recommended that the TR122 module and a controller be installed alone in the crate until the user is familiar with TR122 operation enough to integrate it with other modules in an instrumentation system. When the TR122 is installed in a slot where it has not been before, the CPCI system will detect it as a new device at which point it will need access to driver software. The software supplied with the TR122 is designed to operate on platforms with Microsoft Windows 2000 and newer operating systems. To install drivers on these systems, point the installation wizard to the location of the included CD. A CD drive may be located on the crate controller or it may be mounted over a network connection using the wizard's "specify a location" option. The files the installation wizard will need are located in the root directory of the CD. Follow the steps displayed by the installation wizard at the end of which the wizard should indicate that a "TR122 Transient Recorder" has been installed.

To install the TR122 software, use the Windows explorer or similar tool to locate the software CD. Point to the directory on the CD identified as \TR122\Install. Run the program SETUP.EXE and follow the on screen instructions for the familiar software installation process. When any new software has just been installed, it is usually best to reboot the crate controller before attempting to use the software, even when this is not strictly necessary. At this point the installation is complete and the TR122 and turnkey software are ready to be used.

1.3 Front Panel

The TR122 front panel provides three standard BNC connections to the instrument electronics. These are labeled for the two channels and the external trigger. There is also a serial number label on the card ejector lever. This number will be presented by the TR122 software to indicate which TR122 is being used, which may be important when there are multiple such instruments in the system.

While the instrument has some immunity from typical connection mistakes, care should be still used when connecting external devices. Always observe the maximum ratings indicated in the specifications. Also note that some of these ratings indicate particular settings. For example, the 50 ohm termination is only capable of handling 4 watts, thereby limiting the peak voltage that can be handled with this setting.

The external trigger input is limited to $\pm 10\text{V}$ peak. While slightly higher voltages will not damage the unit, they will operate a protection mechanism. It is recommended that care be taken to keep this input within its specified range. This input is typically used to provide an effective third input which may carry only binary trigger information or to provide some ability to synchronize multiple instruments.

2.0 Using TR122 Software

The TR122 comes with a turnkey software package which allows the user to quickly take advantage of all instrument features in a standalone configuration. This software is designed to coexist with other instruments as well as to allow multiple TR122 instruments to be installed and operated in the same CPCI crate. Installation of the software is discussed in section 1.2 above. The software is designed to have the look and feel of an oscilloscope to take advantage of user familiarity with such instruments, while supporting features unique to a transient recorder. This discussion assumes that the user is familiar with the operation of an oscilloscope.

Upon starting the TR122 software, a splash screen will be displayed while the software searches for and configures a TR122 device in the system. The software will locate a TR122 which is not already in use and complain if one can not be found. Assuming a TR122 is found, a clicking of relays can be heard as the software configures the unit to default settings. The control panel screen shown in Figure 2.1 will appear and its title bar will indicate the serial number of the TR122 which is being operated. This serial number corresponds to that found on the card ejector lever of the instrument. To reconnect this instance of the control panel to another TR122 instrument in the system use the *Select* item under the *Device* menu. The software prevents more than one control panel instance from addressing a single TR122 instrument.

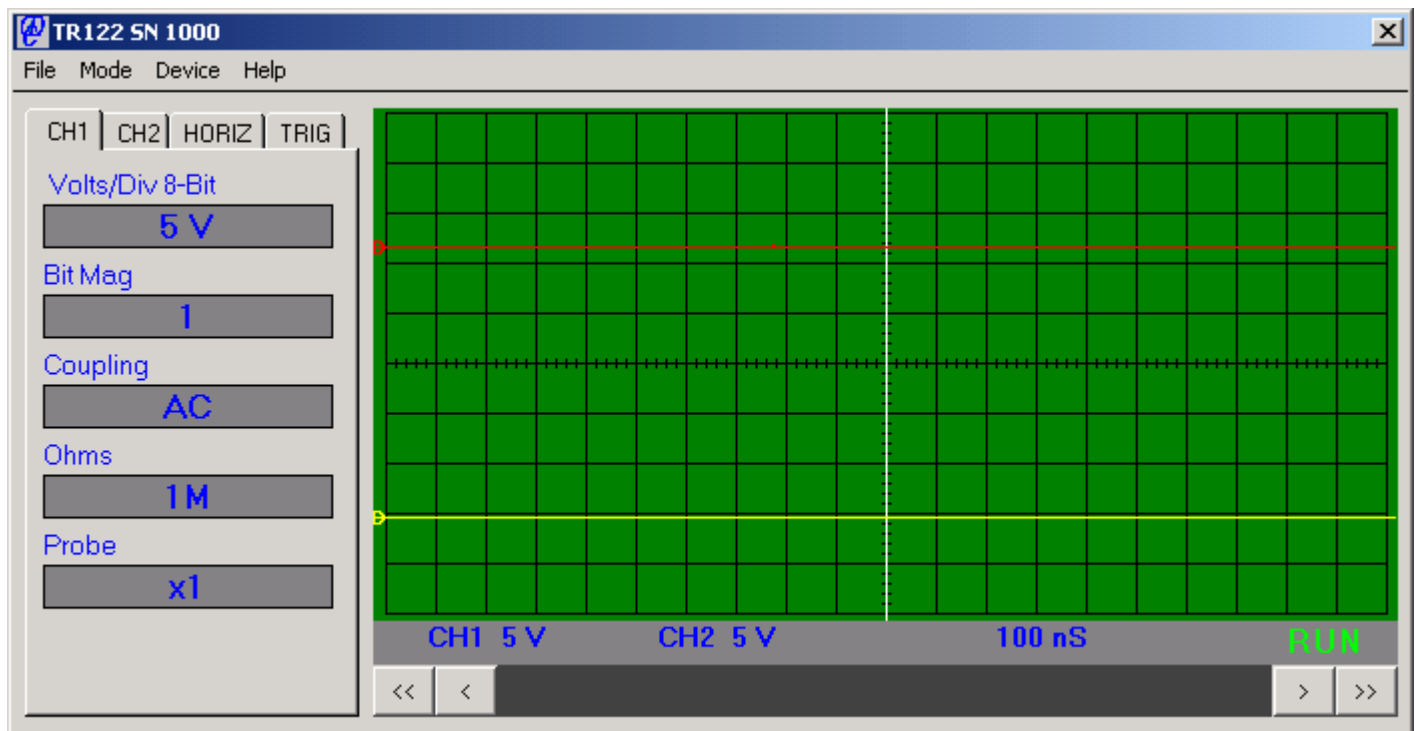


Figure 2.1 TR122 Control Panel

The TR122 starts in the oscilloscope mode and returns to default settings for all controls. This will also be the case any time a new card, or the same card, is found using the *Select* menu item. This gives the user a practical starting point from which to configure the measurement environment. With the factory default display configuration, a real time display of data from both channels will immediately appear in the oscilloscope window and a vertical white line will indicate the time location of the trigger event for each record.

2.1 Customizing The Display

Data from the TR122 are displayed in an oscilloscope display format. The colors and amounts of information displayed can be adjusted with a high degree of freedom. It is useful to review the configuration at this point to become familiar with what is being displayed. To configure the display, double click the left mouse button on the oscilloscope screen. The dialog window in Figure 2.2 will appear. Changing any item in the dialog will immediately affect the display.

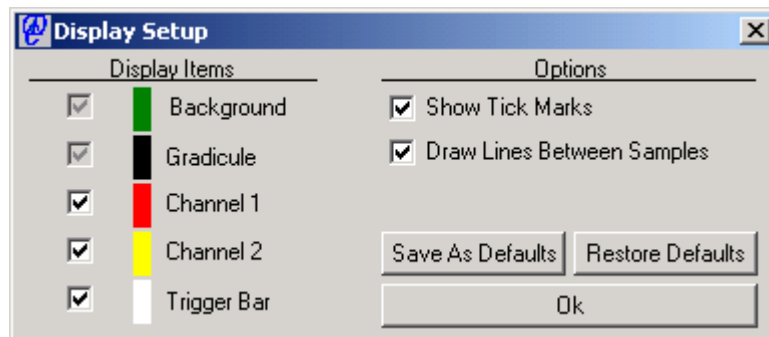


Figure 2.2 Display Configuration Dialog

The *Display Items* list represents all of the information which can be displayed on the oscilloscope screen. The check boxes indicate which items are being displayed and the color bars are the colors in which they are being displayed. Some items can be removed from the display. For example, to view only one channel simply uncheck the other channel by clicking on the check box. To change the color, click on the color bar for an item to display a standard color selection dialog. Clearly, if the background and the gradicule are the same color, the gradicule will effectively disappear. The same would be true for any other item.

The options list allows for selection of optional display features. The *Show Tick Marks* control toggles the display of the tick marks on the center gradicule lines. *Draw Lines Between Samples* places a best fit line between each sample on the screen making the display bright and easy to read. In some cases it may be advantageous to see the discrete sample points by unchecking this option.

Use *Save As Defaults* to cause these settings to be loaded each time the software is started. These settings are saved in a file called *TR122disp.cfg* in the same directory as the software executable. This file is not human readable. To restore the factory defaults, this file should be deleted. The *Restore Defaults* button will cause the contents of this file to be loaded or the factory defaults will be loaded if this file is not present. The *Ok* button just closes the dialog.

2.2 Oscilloscope Mode

Keep in mind that a transient recorder is not an oscilloscope. Although the TR122 software makes a valiant effort at imitating an oscilloscope, the user is bound to be disappointed in comparison to instruments designed for this purpose. A transient recorder is somewhat like a deep memory oscilloscope restricted to operate in the single shot trigger mode. The TR122 oscilloscope mode uses an independent software thread to record and display transient records of the entire two million sample memory at a rate dependant on the selected horizontal time base. To achieve slower timebases the software must lower the sample rate and compress the data. At faster horizontal settings this is fairly undetectable, while at slower settings the transient recorder characteristics can be seen as full memory records are collected and displayed. On the other hand, significant time frames of data are always available, making it easier to capture activity well before or after the trigger.

The oscilloscope mode is intended to be used to establish the settings required to perform a transient measurement. Vertical and horizontal selections along with trigger source selections have an immediate effect on how the data is being acquired and displayed. The user is likely familiar with most of the controls and settings provided. The typical vertical, horizontal, and trigger menus of an oscilloscope are accessed by clicking the tabs at the upper left of the window. The following discussion provides some additional insight into how these controls are used.

2.2.1 Vertical Settings

The vertical controls for each channel are under the tabs marked *CH1* and *CH2* as shown in Figure 2.1 above. The oscilloscope screen in any mode always displays only 8-bits of the 12-bit data stream provided by the TR122. The *Volts/Div 8-Bit* setting assumes that the software will be concerned with only the upper 8 bits. Of course, when data are eventually saved to disk in transient mode, all 12 bits will always be represented.

To allow the user to examine what is happening within the lower bits, the software provides a setting called *Bit Mag*. This can magnify the image to 1,2,4,8,or16 by sliding the visible bits down the sample by 0 to 4 bits. It should be noted that the offset and trigger settings are calibrated to be valid with the 8-bit display and will not be coherent with the *Bit Mag* set to greater than 1. The volts per division display below the oscilloscope screen will always display a value taking the *Bit Mag* into account. The volt per division display at either location can be used to adjust the front end attenuator of the TR122.

All controls in the TR122 software can be adjusted by clicking on them with the mouse buttons. For most, though not all, controls the left mouse button will increase the value of the control and the right mouse button will return its value.

The coupling, input impedance, and probe settings are as they would be on an oscilloscope. The probe setting has no physical effect on the instrument, but simply multiplies the displayed values for volts per division for ease of use when an oscilloscope probe is connected to the instrument.

The analog offset is typically adjusted as part of the vertical controls of an oscilloscope, known generally as vertical position. With a transient recorder it is more typical to discuss offset. The term offset will be used in the context of the TR122. The offset allows a DC value to be added to the input signal, typically to bring a signal with a DC offset within range of the instrument; but also to allow for separation of multiple signals on a display, such as would be a typical use of an oscilloscope vertical position control. Within TR122 software the offset is controlled directly on the screen with the mouse. The arrow icons at the left of the screen indicate where the offset is for each channel by color. Move the mouse cursor to the left most horizontal division of the oscilloscope screen and click the left mouse button to move the offset for the nearest channel to the screen position of the mouse. Holding the left mouse button and moving the mouse will drag the location of the offset. The position of the icon represents the display location of a zero volt DC signal input. The data behind the value, such as that stored to disk, will include the DC offset.

2.2.2 Horizontal Settings

Selecting the *HORIZ* tab in the controls dialog at the left of the oscilloscope screen will display the horizontal settings controls as shown in Figure 2.3 below. In oscilloscope mode the horizontal *Sec/Div* control is the only enabled control. This control is similar to the control of the same name on an oscilloscope. The setting can also be adjusted by clicking on the status line copy of this value located below the oscilloscope screen.

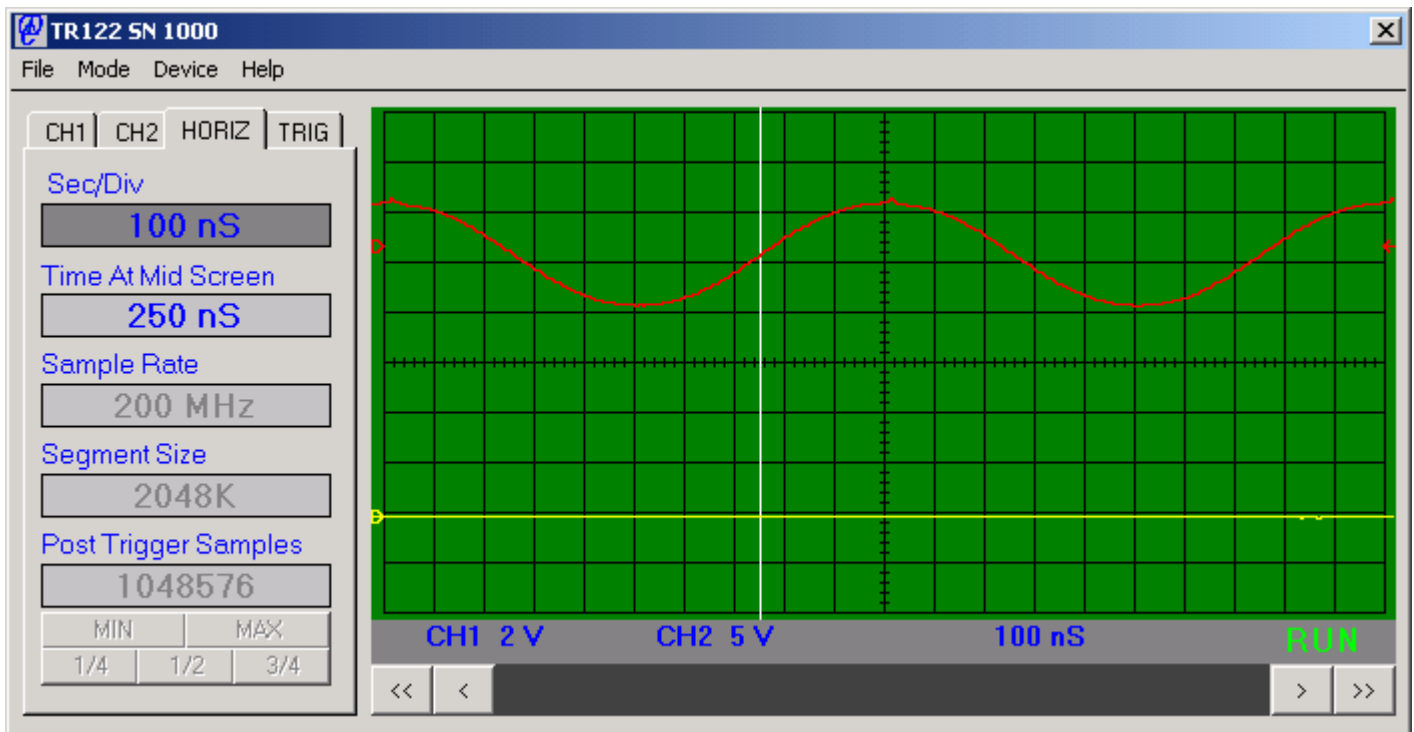


Figure 2.3 TR122 Control Panel Horizontal Controls

The horizontal menu includes a status display value in *Time At Mid Screen*. This box will display the time represented at the horizontal center line of the gradicule with respect to the trigger location. The units and resolution will depend on the *Sec/Div* setting and the magnitude of the value. This display can be quite useful when the trigger location is not displayed or is beyond the horizontal bounds of the screen. Because the full size of the memory is quite large, it is possible to view events substantially on either side of the trigger and know where they are in time. This feature is available in transient recorder mode as well. For oscilloscope users, this feature starts to show the strengths of a transient recorder design for certain types of measurements within the same price/performance class of instrument.

The horizontal location at which the trigger bar is displayed can be adjusted by clicking and holding the left mouse button on the display and dragging the display in either direction. The position can also be adjusted in steps with the < and > buttons at the bottoms of the screen. These controls are similar to the horizontal position control of an oscilloscope.

The horizontal menu in the oscilloscope mode also shows the ghosted values of the transient recorder settings being used. The memory geometry will be constant in oscilloscope operation, but the sample rate will change with certain ranges of the *Sec/Div* setting. This display can be useful in determining what sample rate setting may be required to perform transient measurements on a given input.

2.2.3 Trigger Settings

Selecting the *TRIG* tab in the controls dialog at the left of the oscilloscope screen will display the trigger settings controls as shown in Figure 2.4 below.

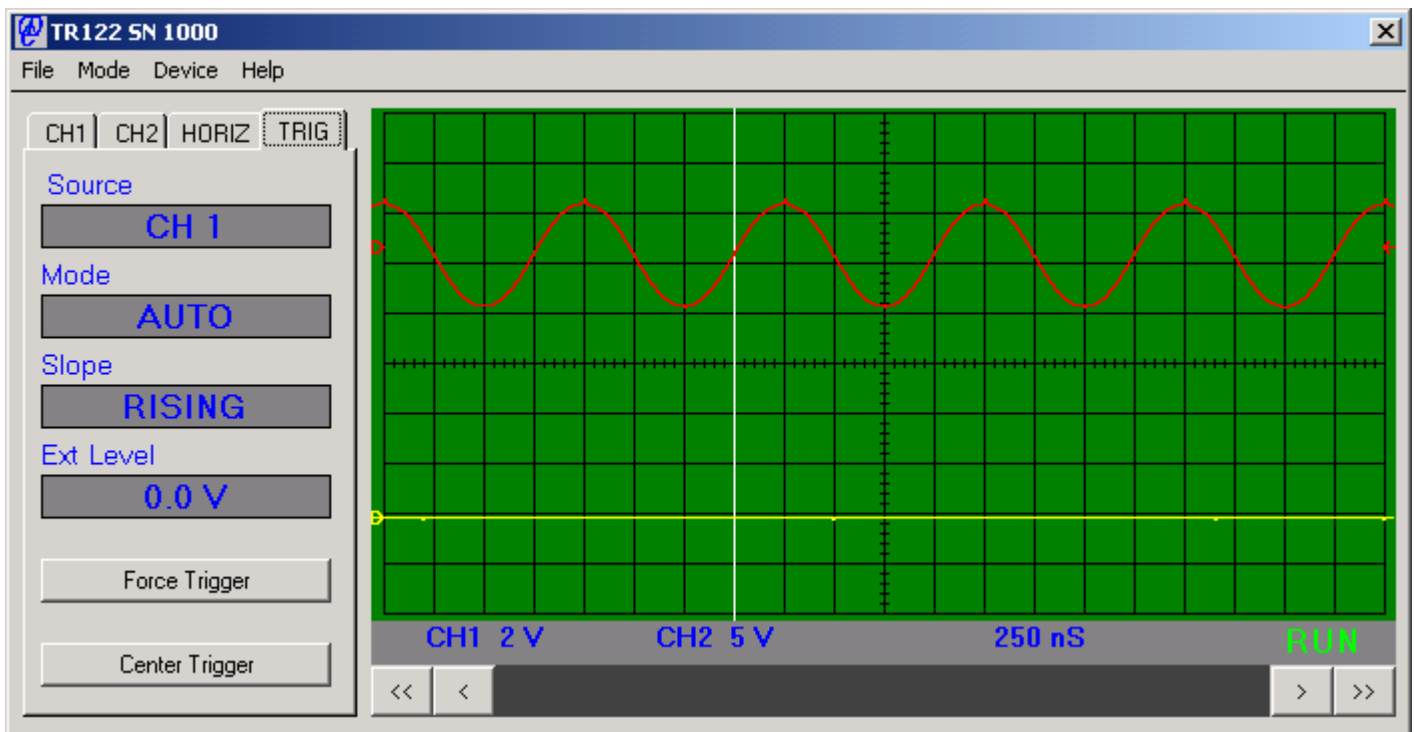


Figure 2.4 TR122 Control Panel Trigger Controls

In oscilloscope mode the trigger controls mirror those on an oscilloscope. The trigger *Source* can be either channel or the external trigger input. The trigger *Slope* can be rising or falling edge. The trigger system can operate in a *Mode* of AUTO, NORMAL, or SINGLE which mirror their oscilloscope equivalents. The NORMAL mode will record and display new information only after receiving a trigger event and again after each subsequent trigger event following complete display of the data from the previous event. The SINGLE mode will display information only from the first such event after which the *Run/Stop* control in the lower right corner of the screen will be set to STOP. Data from an additional trigger event can only be recorded by clicking on the *Run/Stop* control to set it back to RUN. The AUTO mode behaves like the NORMAL mode except that the software will inject a trigger if one is not received in an internally determined time. This trigger will of course not be synchronous with any event in the input signal but the resulting display can be useful in the event that it is not clear how the trigger source should be prepared. If a trigger event does occur before the internal timeout on each cycle, the display will be properly synchronized to this event.

The *Run/Stop* control can be set to STOP in any trigger mode to cause the software to ignore any additional trigger events. The display will be a static representation of the last captured data. When the trigger system is set to STOP in the oscilloscope AUTO or NORMAL modes, only the data already on the screen will be available. When the SINGLE trigger mode sets the control to STOP, the entire memory is available for review. This distinction relates to the ongoing recording operations in the other modes. That is, the SINGLE mode does not record pre trigger samples for an anticipated subsequent trigger while the other modes do, which would of course overwrite data from any previous trigger. When switching between trigger modes, the *Run/Stop* control is automatically returned to RUN to avoid the typical need to do this manually.

The analog voltage threshold at which a trigger is detected is called the trigger level. For the external trigger input, this level is adjustable in an absolute range of -10V to $+10\text{V}$ using the *Ext Level* control. If the trigger source specifies that one of the channels should be used, the range becomes somewhat larger than the full scale analog input range. The location of the trigger threshold will be indicated in an abstract way with an arrow on the right side of the oscilloscope display with the color of the selected source channel. The level can be adjusted by clicking the left mouse button on the oscilloscope screen within the far right horizontal division. The trigger threshold will be moved to the location of the mouse cursor. The location can be dragged by holding the left mouse button down while moving the mouse. If the offset were to be adjusted, the trigger location will be further adjusted with it to keep the intended trigger threshold with respect to the new zero volt line. This may force the trigger location off the screen, which will be indicated by a vertical pointing arrow at the far left top or bottom of the screen. This is the benefit of a trigger range greater than the full scale analog input range.

The *Force Trigger* button allows the user to issue a software trigger at any time in any mode. This function is independent of anything else in the software. If the instrument and software are configured to respond to a trigger, clicking this control will cause it to act the same as if the trigger had come from the specified source. Of course there is no way to synchronize this action to any event, but this control can be useful in performing setup operations, particularly in transient recorder mode.

Finally, the *Center Trigger* button automatically returns the display settings so that the trigger event is displayed at the horizontal center line of the gradicule. This can be useful when exploring the outer reaches of the recorded time in any mode and it is desired to return to time zero, the time of the trigger event.

2.2.4 Saving Waveforms And Settings

In the oscilloscope mode there is generally not a known set of data in memory, except perhaps as the result of a single shot trigger operation. The transient recorder mode handles the recording to known data sets in memory somewhat like a sophisticated single shot trigger operation. Therefore, saving of sample data to disk is not supported in oscilloscope mode. However, there is always a waveform image on the display and saving this image to disk is supported. Select the menu item *Save Image* in the *File* menu. A standard file dialog will ask for a file name under which to store a bitmap image of the oscilloscope display. This bitmap can be used in any document such as shown in Figure 2.5 below. When the image is saved to disk, the *Run/Stop* control is set to STOP. If the control was set to RUN, it is returned to RUN when the file save is complete. This time will be very brief.

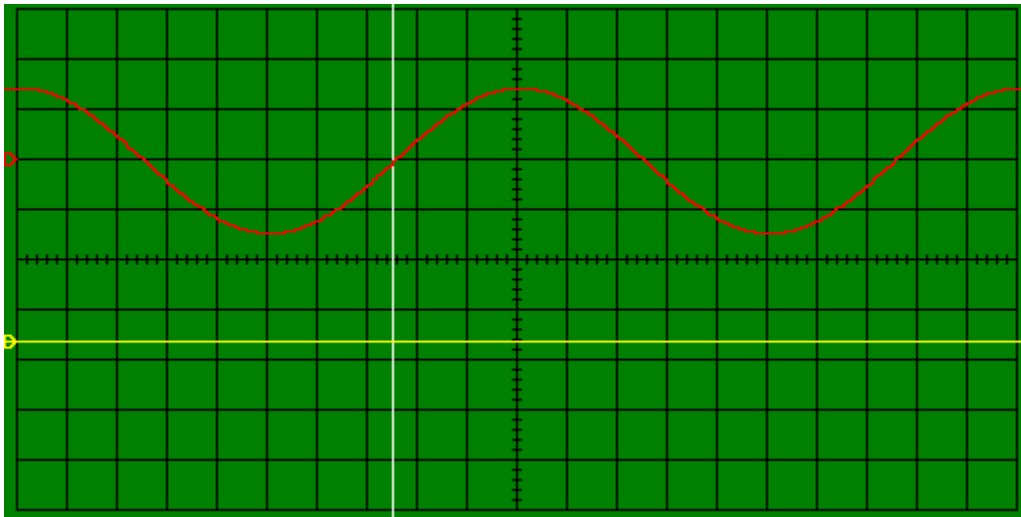


Figure 2.5 Captured TR122 Oscilloscope Screen Image

Generally, the oscilloscope mode is used to setup the TR122 instrument for transient recording operations. Some setting can not even be changed once transient recorder mode is entered. For this reason, if settings are to be saved to disk to preserve a given setup, the oscilloscope mode would be the right place to do it. The menu items *Save Settings* and *Load Settings* are available in the *File* menu in the oscilloscope mode. The *Save Settings* item will prompt for a file name under which to save the current settings of the user controls. When the TR122 software is started, default settings are always installed. Stored settings can be retrieved using the *Load Settings* menu item.

2.3 Transient Recorder Mode

Changing between oscilloscope and transient recorder mode is performed by selecting a mode from the *Mode* menu item. This menu indicates the current mode with a check mark. Starting the transient recorder mode will result in a blank display screen and the *Run/Stop* control indicating STOP. The status display of *Sec/Div* will be missing as this value does not have meaning until a recording is made and displayed.

The vertical controls under *CH1* and *CH2* perform the same as they do in the oscilloscope mode. The trigger controls under *TRIG* also perform much the same except that the trigger modes of the oscilloscope do not have any meaning in transient recorder mode. As a result the trigger *Mode* control is disabled.

The offset and channel source trigger level controls are not adjustable in the transient recorder mode. These settings are generally adjusted while the user has a good look at incoming data in the oscilloscope mode and then are not to be changed while a transient measurement is taken. All other controls can be adjusted. However, adjusting any control will invalidate any previous transient record and the display screen will be cleared. It is important to recognize that a transient record is a once and done event with a very specific set of parameters on the instrument. By operating in this manner, the user can be sure that any transient records displayed or written to disk were in fact taken with the settings shown. In contrast, the oscilloscope mode, which also clears the screen on a setting change under most conditions for the same reason, will generally refresh the data with a new record following shortly after the setting change. The transient recorder mode requires user intervention to collect and display each record.

2.3.1 Configuring Transient Record Memory

Beyond the basic setup which matches that of an oscilloscope, configuring the transient recorder mode involves selecting a sample rate and memory geometry. Defining this information is a rather manual step which is unique to a transient recorder, but it allows the user to control and be aware of specific parameters of the recording. These parameters are critical knowledge for some types of measurements and are generally not reported by an oscilloscope. To allow control of these parameters, the *HORIZ* tab control in transient recorder mode will appear as in Figure 2.6 below.

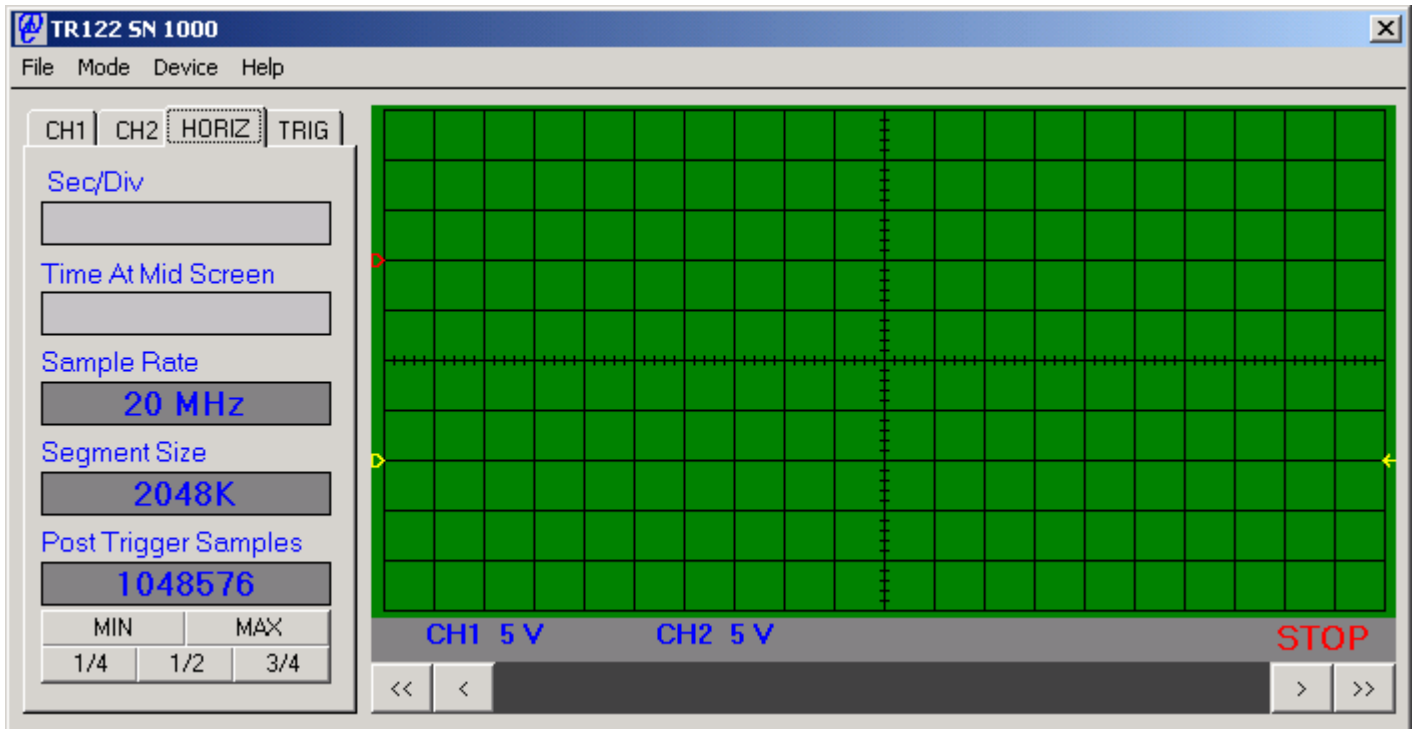


Figure 2.6 TR122 Memory Geometry Controls

The all important *Sample Rate* control is self explanatory to transient recorder users. This control will generally be adjusted based on the frequency characteristics of the input signal being viewed and by trading off recorded signal detail with the length of time which can be stored in the instrument sample memory. When selecting a sample rate (measured in MHz or Million Samples Per Second (MS/s)), keep in mind that the bandwidth of the TR122 analog circuitry is always 100 MHz, and the roll off of the pass band is not steep. Any incoming signal with a frequency component greater than half the sample rate can potentially appear misleading when displayed due to aliasing, an important digital signal processing mathematical concept which must be understood for successful use of a transient recorder or a digital oscilloscope. Because the input bandwidth is 100 MHz, the potential for aliased information in the display is limited when the full 200 MS/s sample rate is used. However, the effect can become pronounced at lower sample rates.

The time recorded in memory will be equal to the number of samples stored in memory (samples) divided by the sample rate (samples/second). The full depth of the memory is two million samples. Therefore, at the full sample rate of 200 MS/s the memory will hold 2M/200MS/s or 0.01 seconds of time. At the slowest sample rate of 1 MS/s this expands to 2 seconds of time. The user is probably aware that both times can be considered fairly long transient events.

In some cases it may not be necessary to record so much time. It may also be the case that several events are expected to occur in rapid succession. It is possible to configure the TR122 memory to capture multiple trigger events by dividing its full memory into smaller segments. The *Segment Size* is measured in thousands of samples (K samples). The smallest available segment size can be referred to as a *block* of memory. The block size of the TR122 is 64K samples. Note that all dimensions suffer from the binary number system employed by the modern computer, so block and segment sizes will be powers of 2. For example, 64K samples is 65536 samples. A segment in the TR122 can be sized as a power of 2 times the block size, up to the size of the full memory. That is, the segment size can be selected as 64K, 128K, 256K, 512K, 1024K, or 2048K samples. The memory can therefore be divided into 32, 16, 8, 4, 2, or 1 segments respectively. Thus, for short rapidly occurring transient events, up to 32 such events can be stored in memory before the data must be read or a new recording started.

When a recording is started, the TR122 will immediately begin recording data in the first segment of memory, filling the segment with samples and returning to the beginning of the segment to start again in a continuous loop. When a trigger event is detected, the TR122 will begin counting samples to be recorded after the trigger event, as defined by the user in the control *Post Trigger Samples*. When this number of samples has been recorded, the TR122 moves on to the next segment and begins again recording pre trigger samples.

Clearly for sample data in this circular buffer architecture to be sensible, the segment must be fully written with pre or post trigger samples related to the trigger event. To guarantee that this happens, the TR122 will not be sensitive to a trigger event until the segment has filled at least once with pre trigger samples. Once a trigger is received, the TR122 will not be sensitive to another trigger until all post trigger samples have been recorded and the next segment has been filled with pre trigger sample data. Therefore, the minimum time between trigger events will be defined by the memory geometry selected as

Minimum time between trigger events = (segment size / sample rate) + (post trigger samples / sample rate)

Internally to the TR122 software and hardware, the number of *Post Trigger Samples* can be defined to any multiple of 2. However, this resolution is rarely needed. The TR122 control panel allows the *Post Trigger Samples* control to be adjusted in steps of 512 samples. The limits will require there to remain at least 1K (1024) pre trigger samples and at least 1K post triggers samples within the segment. When selecting a segment size, the *Post Trigger Samples* control is automatically adjusted to half the segment size. The shortcut buttons under the control allow setting commonly used values for *Post Trigger Samples* with one click.

2.3.2 Performing A Transient Recording

To prepare the TR122 to capture a transient record, set the *Run/Stop* control to RUN. At that point the memory geometry controls are frozen and the TR122 begins recording pre trigger samples and waiting for a trigger event. The other controls remain available to the user, however it should be kept in mind that adjusting controls after the first trigger is received could cause an inconsistency in the data between one segment and the next.

The TR122 will continue to record data until enough trigger events have occurred to fill all segments, at which point it will automatically stop recording and display the data from the first segment. To stop the recording and display data before all segments have been filled, manually set the *Run/Stop* control to STOP. The data from the first segment will be displayed on the oscilloscope screen with the trigger event at the center of the gradicule.

When a transient record is displayed, the dark bar below the display will be filled with red and green bands. This bar is referred to as the *sample memory profiler*. The green band covers the portion of the memory which is filled with valid samples. The red band lays over top of the green band and indicates which segment of memory is currently being displayed. The size of these bands is relative to the segment size, and of course for geometry consisting of a single full memory segment, the sample memory profiler will be entirely red. This display provides a clear visual indication of how many trigger events were received in a transient record, the relative size of the data for each segment, and which trigger event is being examined. Figure 2.7 below shows the display of a transient record in which four trigger events filled 64K segments before the record was discontinued by the user. The second segment is displayed.

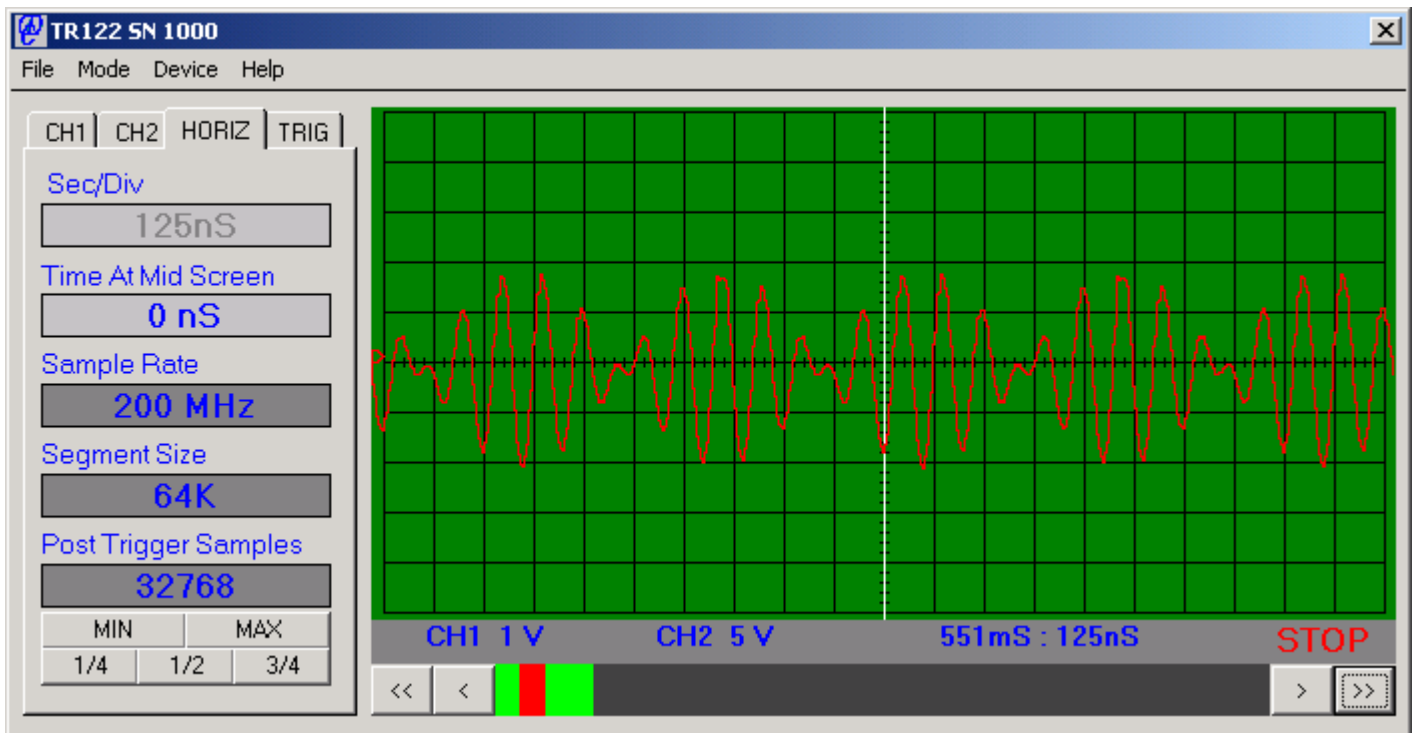


Figure 2.7 Displaying A Transient Record

When a transient record is being displayed, the software will generate a seconds per division status indication in its traditional place. This number is not adjustable. While the oscilloscope mode is designed to compress or expand samples on the screen to reach a specific time gradicule, the transient recorder mode strictly presents 25 samples per division. It is intended to be crystal clear what the pure sample data looks like when graphed at one vertical axis point to one horizontal axis point. Therefore, the seconds per division value will only change based on the sample rate of the transient record.

An additional time point is displayed on the status bar along with the seconds per division. This is the absolute time at which the trigger event for the currently displayed segment occurred with respect to the first trigger event, based on a microsecond counter in the TR122. The display of the first trigger event will always show zero microseconds (0uS) in this location. Of course, the timer could lapse during a transient record and thereby provide strange results, but it is good for over an hour – an eternity in the world of transient recording.

To move the display between segments use the << and >> buttons on either side of the sample memory profiler. This action will set the display to show the trigger location of the new segment at the center of the gradicule. To move the trigger location and explore the reaches of the memory segment before or after the trigger, use the same mouse screen control or the < and > buttons as in the oscilloscope mode display. The *Time At Midscreen* display under the *HORIZ* tab will represent the time at the middle of the display gradicule with respect to the trigger event in the currently displayed segment.

2.3.2 Saving A Transient Record

The sample data in a transient record can be saved in one of three formats. The current displayed image can be saved as a bitmap as in the oscilloscope mode, or the data can be saved in binary or comma separated text format. A bitmap image may be useful for a brief report about a recorded signal. The binary format would typically be used by some form of custom post processing software. The text form is somewhat human readable or it may be used for processing in spreadsheets or other standard post processing software.

Saving a bitmap image follows the same procedure discussed for the oscilloscope mode. Saving data is completed using the *Save Data* item under the *File* menu. This item is only available in transient recorder mode. It will also only work if there is a valid transient record to save. Otherwise it will report an error. A standard file save dialog box will appear with the file type selectable as binary or text in the *Save As Type* selection box of the dialog.

The text data file format will contain a line of text for each sample. The samples are in chronological order from oldest to newest. The resulting file will be quite large and will probably need to be reduced for use in most data analysis programs. Some users may also find it easier to understand this format than the binary format for use in custom post processing software. Each line of the text file is formatted as follows.

<Segment>,<Post Trigger>,<Channel 1>,<Channel 2>

Where: **<Segment>** is the memory segment in the range 0 to 31 dependant on the current memory geometry settings
<Post Trigger> is 0 if the sample was taken before the trigger event and 1 if the sample was taken after the trigger event
<Channel x> is the absolute value of the sample over the current full scale range as an offset binary value of 0 to 4096 – that is, with the zero volt level of input signal plus offset being 2048. This value includes any effect of the instrument offset settings.

The binary file format contains sequential samples in a six byte equivalent to the line of text in the text file format. It is written as a data structure from the C programming language and stored with no padding between samples. Post processing software will need to be aware of this structure's format and will generally read data from the file into an equivalent structure to preserve the byte ordering. The structure is as shown below, assuming the Windows software conventions that a BYTE is an unsigned 8-bit number and a WORD is an unsigned 16-bit number. The descriptions of the elements are equivalent to corresponding items in the text format above.

```
typedef struct binary_data_record_st
{
    BYTE segment;
    BYTE trigger_valid;
    WORD sample_ch1;
    WORD sample_ch2;
} T_BINARY_DATA_RECORD;
```

Note that these data file formats do not contain any information about the settings of the instrument from which they were written. Some information about the sample memory geometry may be inferred, but other potentially important settings information must be recorded separately if needed. The objective of the *Save Data* function is to save raw data for post processing in its simplest possible format. Many post processing programs will require no more data than this, while still others may require little else. Most will require a knowledge of the sample rate or other basic points that the user might consider recording in the file name. All will be affected in complexity and processing time by the size of the data file and anything other than uniformity in its content. This is the logic behind the simple format given.

3.0 TR122 Software Source Code

Instruments such as the TR122 are only as useful as the software which runs them. While the software provided with the TR122 makes the instrument very functional in a wide variety of applications, it is impossible to predict the full range of possible user needs or the software environment in which the instrument may be installed. Data Design has decided that the best way to meet as many user needs as possible is to provide a window into the workings of the instrument. To accomplish this, the source code for the software is published with the software distribution.

This open source approach provides the user with nearly infinite flexibility. It also provides the opportunity for many bad headaches. The inner workings of the software are not for the novice user. Complex tricks and traps exist at every corner with an advanced instrument such as the TR122. Concepts of multistage pipelines, direct memory address, advanced data structures, threads, and other complexities must be understood. It is a challenge even for the most skilled programmer. The good news is that the design is carefully structured and this section of the manual provides a roadmap to the advanced user.

IMPORTANT NOTICE Information on source code and the source code itself is provided AS IS and without warranty or support. Data Design is not able to provide technical assistance to those attempting to modify the source code. It simply would not be possible to provide the instrument at reasonable cost if such support were included in the price. Please review the source code license agreement at the end of this manual before using the source code. That said, however, Data Design does offer design services for hire. Information can be found on the website at the front of this manual. Also, from time to time, additional applications and source code will be posted on the website for free download.

It should be noted that this section of the manual covers the workings of a particular code base for a particular instrument. Any such software works with a hardware-software interface defined in part by the hardware itself. At the simplest level this interface amounts to writing various command and configuration registers and reading various data and status registers. Still, understanding many of the complexities found internal to the code base might require a working knowledge of the hardware. The hardware itself is not discussed in this manual and is considered somewhat proprietary to Data Design. While it will become evident in this discussion that the hardware could also be modified to meet additional unique and complex requirements, this is substantially beyond the scope of most users' needs and abilities. As such, to maximize the usability of the open source environment, the interface to hardware is left somewhat abstract in order to simplify the discussion. This understanding should be kept in mind when reviewing the source code and deciding what should or should not be modified.

While there is some discussion of hardware details within the source code itself, the discussion in this manual is aimed at common reasons why an advanced user would require access to the source code. These include such tasks as interfacing to a custom top level application, connecting the instrument to an existing software environment, or using the instrument in another operating system. As such, the discussion of the top level application presented in section 2 is also limited, though the source code for that application is also provided.

3.1 TR122 Software Architecture

The hardware and software design of the TR122 is rooted in well accepted flexible, layered design concepts. The functional stack of all elements in the instrument design and data flow are shown in Figure 3.1 below.

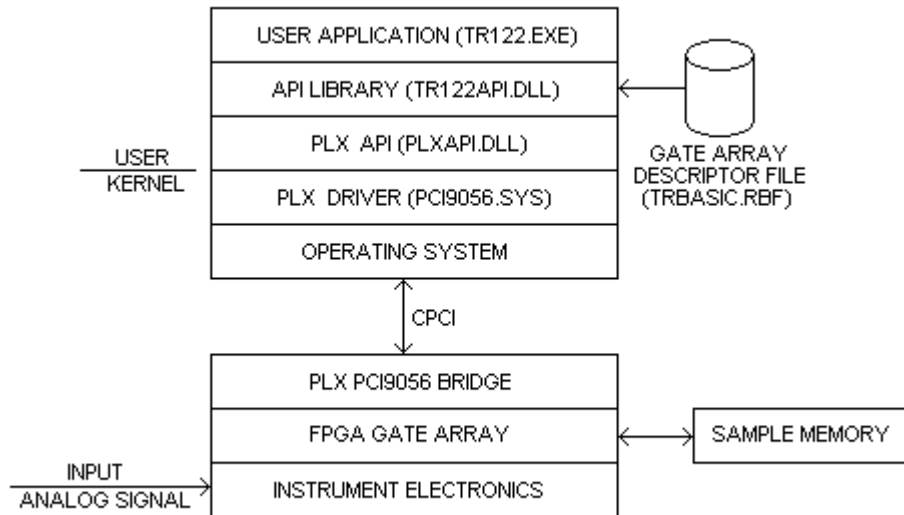


Figure 3.1 TR122 Software Architecture And Data Flow

The instrument hardware is mostly defined by a design which is software loadable into a large field programmable gate array (FPGA) on the main board. The binary image of this hardware design is saved on disk in the host computer and loaded to the instrument when the software is started, or more specifically when the instrument application programming interface (API) library is loaded in memory. The FPGA hardware design contains functional blocks to interface with the instrument electronics, sample memory, and the PCI interface. Where the software addresses specific registers on the instrument hardware, these registers are located within the FPGA hardware.

The PCI interface itself contains a popular PCI bridge from PLX Technologies. The advantage of using this device is that the gate array itself can be soft loaded and even potentially reloaded after the host system is running. This provides flexibility and the ability to apply future updates and enhancements to the hardware without the need to return the instrument to the factory. The PCI9056 also has support available for a number of operating systems, including native kernel support in some popular operating systems such as Linux. Under Windows a comprehensive set of drivers and API features takes a lot of the work out of commonly needed operations from basic device access to direct memory address support. The TR122 software distribution includes an installation information file which directs the installation wizard to install the Windows WDM kernel level driver and the associated PLX API for the benefit of a newly installed TR122 instrument. These driver and API files contain the code necessary to work with the operating system and the bridge chip on the other side of the PCI bus. They are provided by PLX Technologies for distribution with systems that use these bridge chips. When working with the software at this level under any operating system it may be helpful to acquire a PLX Technologies software development kit. Generally, because the instrument API is

developed well within the user layer, it will not be necessary to use any kernel level driver development kits. However, some operating systems may draw this line at other points.

The TR122 API library contains code which performs the most difficult functions required to operate the instrument and exposes to the layer above a comprehensible set of calls which perform abstract functions related to a transient recorder and specific TR122 hardware. The API is provided as a dynamic link library (DLL) written in the C programming language. This is where the bulk of the TR122 software is located and is the primary subject of this section of the manual. The API is designed to relieve the top level application from even the most rudimentary house keeping duties. When the library has been successfully loaded by a top level user application, the TR122 will be configured, calibrated, and ready for use.

The top level user application is left with very little substantial work to do. This is the layer which provides all the fancy features – the buttons, lights, dialogs, and message boxes. In the TR122 turnkey software package, the top layer is written in Microsoft Visual Basic 6 (VB). It could have as well been written in National Instruments LabView, Borland Delphi, Microsoft C, or any other environment which can access a dynamic link library as an API. The source code for this application is included in the distribution but is not discussed in this manual. It may be instructive to review the source code in the Visual Basic environment. The program begins by loading the main form, aptly called Form1, which will automatically load the API library.

3.2 The TR122 API

The TR122 application programming interface (API) is provided as a Windows dynamic link library (DLL) which is installed in the Windows system directory (\WinNT\System32 or equivalent). Also stored at this location is the binary image file for the gate array in the instrument. When the API DLL (TR122API.DLL) is loaded by the application, or by the operating system on behalf of the application, initialization code is executed which finds a module, loads the gate array, and sets system defaults.

The API exports functions which provide an easy path to use features of the instrument. Exported functions, data structures, and definitions which will be needed by an application calling the API are defined in the TR122API.H header file. All exported functions are defined with a name prefix of TR122_ and will always return an unsigned 32-bit number which will be set to 1 if the function performs as expected and 0 if a difficulty is encountered. Any substantive data returned by a function is returned by reference in the arguments.

Several steps are taken to improve compatibility with a wide variety of programming environments. An export definition file (TR122API.DEF) is part of the source code so that exported names are explicitly defined for the benefit of environments which may not be aware of Microsoft name mangling standards. All arguments of exported functions are atomic types and pointers to atomic types as opposed to complex data structures which may be complicated to define in some environments. The source code is written in the C programming language with some Microsoft conventions geared towards the Windows programming model for the Intel architecture. The following are some useful relationships between data types for arguments in several software environments.

API Argument Type	Description	VB Type	LabView Type
BYTE	Unsigned 8-bit number	ByVal Byte	uInt8
DWORD	Unsigned 32-bit number	ByVal Long	uInt32
long	Signed 32-bit number	ByVal Long	int32
char*	Pointer To 0 Terminated Text String	ByVal String	CStr

There are several points to consider regarding range of arguments. Any type may also be presented in an argument as a pointer to data of that type, while the char (strictly a signed 8-bit number) always appears as a pointer in the API and refers to the location of a null terminated text string. Generally it is an error for null pointers to be passed in arguments. Some environments required more work than others to avoid passing null pointers to strings. The BYTE data type is sometimes used as a number, but is often used as a proxy for a boolean type where 0 is false and anything else is true. The API avoids the use of 16-bit words of any kind because some environments do not support them or do not support an unsigned version. The use of the long integer type implies that the API is expecting a signed number. The DWORD data type implies that the API is expecting an unsigned number. However, in the later case the range will in most cases be less than 31 bits so that environments such as Visual Basic (VB) which do not have an unsigned 32-bit number can use a signed 32-bit number in its place.

Most advanced programming environments provide an interface to DLL's. However, all have peculiarities which must be understood for successful use. For example, in the VB environment an argument passed to the API is considered a pointer to a data type unless it is noted to be passed 'ByVal'. The VB type 'String' is a pointer, so the argument would be a pointer to a pointer unless the ByVal qualifier was applied to make it a pointer to a character array. The definitions of the API functions as required by VB can be found in the source code for the TR122 top level application.

The exported function list contains several classes of functions. These include abstract functions which contain identifiable hardware and user features of a transient recorder – such as setting a sample rate or issuing a trigger. Another set of functions deals with calibration of the instrument, for use in operations covered in section 4 of this manual. Finally, a set of functions performs operations on an internal device independent bitmap (DIB) which forms the basis of the oscilloscope display screen.

In addition to exported items and others enumerated in the header file, the source code contains a number of private functions and data variables. For the most part the private data are unique to a particular process. A process is an instance of the top level application, such as the user application for the TR122 instrument in a particular CPCI slot. For example, each instance of the top level application will have its own screen DIB and its own settings. If data are to be shared between processes, the API must take advantage of operating system resources for that purposes. For example, each process must be aware of which TR122 instruments are already in use by another process. In addition, a number of operating system semaphores and message flags are used to communicate between processes and internal threads.

Even most advanced users will use this source code as a reference to build additional higher level applications. It would be unusual to actually modify this source code. However, the source code can be a valuable reference in understanding how each of the functions behave. There is extensive documentation in the source code itself. This manual will provide a roadmap of exported functions, an overview of their intended use, and how they fit in to the overall software design.

3.2.1 Calibration Functions

Calibration information for the TR122 instrument electronics is stored in a nonvolatile memory on the module as a series of gains and offsets. The gain for each channel in each front end attenuator setting is stored. There is also a gain and an offset associated with the offset circuitry for each channel in each front end attenuator setting. These settings are only changed by a user during periodic calibration. However, software such as the TR122 turnkey application must contain a method of doing this. These functions are declared as follows.

```
DCAPI TR122_CAL_SetGain(BYTE Channel, long IncDec);
DCAPI TR122_CAL_SetOffsetGain(BYTE Channel, long IncDec);
DCAPI TR122_CAL_SetOffsetOffset(BYTE Channel, long IncDec);
DCAPI TR122_CAL_Save(void);
DCAPI TR122_CAL_Load(void);
DCAPI TR122_CAL_Clear(void);
DCAPI TR122_CAL_Alignment(BYTE AlignmentState);
DCAPI TR122_CAL_SerialNumber(DWORD *SerNum, BYTE OverWrite);
```

Because the TR122 turnkey application is available, it is hard to imagine a situation where a programmer would have to include these functions in an alternative environment. Probably such things should in fact be avoided. However, the API itself makes use of these functions to load calibration information to variables associated with this code and uses these variables throughout the API in normal operation.

One function that might be used regularly is TR122_CAL_SerialNumber. This function will return by reference the serial number written into nonvolatile memory on the module. Be sure to set the *OverWrite* argument to 0 (false). If the serial number has not been corrupted by misuse of this function, the value returned will be the number on the front panel ejector handle of the module.

3.2.2 Abstract Instrument Functions

The abstract instrument functions provide a means to perform a set of operations which are recognizable and expected for a transient recorder. There are some subtle differences between operation in transient recorder and oscilloscope modes which affect these routines. From the point of view of the API, the oscilloscope mode is defined if the 'Auto Thread' is running and transient recorder mode is defined if it is not. The 'Auto Thread' is discussed later in this manual. This discussion will focus on common functionality. Presented below are the exported functions with a brief description of their place in the design. They are ordered differently than in the source code for clarity of discussion.

DCAPI TR122_OpenNext(void);

When the library is loaded, the API will automatically use this function to open and reserve a path to the next available TR122 instrument in the CPCI crate. Therefore, generally the programmer will not have to call this function. If there are more than one TR122 instruments present in the crate, this function can be used to close the current connection and open a connection to the next instrument in the crate as defined by PCI slot number. If no more instruments are found, a connection to the original TR122 will be opened again. If no TR122 is found at all, this routine returns 0, which would have caused the library load to fail and a user warning message box to be displayed.

DCAPI TR122_StartHostServiceMode(void); DCAPI TR122_StartRecordMode(void);

The TR122 hardware has two distinct operating modes called *Host Service Mode* and *Record Mode*. Generally, to update any configuration or to read the sample memory, the TR122 must be in host service mode. A TR122 in record mode will be recording samples in a segment of sample memory. Calling these routines switches the hardware between these modes and performs certain software and hardware housekeeping associated with these changes.

DCAPI TR122_SetSampleRate(DWORD Rate); DCAPI TR122_GetSampleRate(DWORD *Rate);

The sample rate can be set when in host service mode. The sample rate value in the set routine is an internal control value and must be one of the SAMPLE_RATE_XX values found in header file. The sample rate returned by reference in the get routine is a human readable number of megahertz (million samples per second).

DCAPI TR122_SetSampleSegmentSize(BYTE SegmentSize, DWORD PostTriggerSamples);

The sample memory geometry may be set when in host service mode. The segment size sent to the routine is an internal control value and must be one of the SAMPLE_SEGMENT_XX values found in the header file. The post trigger samples is just an integer value but must be an even number due to hardware limitations.

DCAPI TR122_GetStatus(BYTE *Status);

The status returned by reference in this function is defined in the header file. It informs the caller if the TR122 is in host service mode. If the instrument is recording, the status indicates if the memory is full. A caller may poll this routine periodically to see if a transient recording has stopped on its own due to a full memory.

DCAPI TR122_SoftwareTrigger(void);

This routine issues a trigger without restriction just as if it came from any other source. The software trigger is logically ORed with any selected trigger source. The trigger will only affect operation if the TR122 is waiting for a trigger at the time. This trigger source is not synchronous to anything.

DCAPI TR122_GetTriggerInformation(DWORD *NumberOfTriggers, DWORD *Times);

This routine is called to validate a transient record and to determine the amount of data present. It is called internally to the API as part of a display function. However, it can be called from a higher level application. If a valid transient record is present in memory, this routine will return the number of segments filled with valid data and an array of up to 32 time values at which the triggers occurred with respect to the first trigger. The first element in the array will always contain a 0. The times will be a count of microseconds. The time values are full 32-bit numbers, so as an exception to the general rule, to take advantage of the full range the calling program will need to be able to interpret unsigned 32-bit numbers.

DCAPI TR122_SetTrigger(BYTE Source, double Level, BYTE Edge);

The trigger source parameters are set by this function. The source can be 0, 1, or 2 for external, channel 1, or channel 2 respectively. The level can be -10 to +10 for source 0 and -100 to +100 for source 1 or 2. The trigger level for the channel sources is similar to percentage of full scale above or below midscale. The edge argument is 0 for falling and 1 for rising edge sensitivity.

DCAPI TR122_SetOffset(long AbstractOffset1, long AbstractOffset2, BYTE Update);

The offset in electronic terms is another analog signal added to the incoming signal being measured. Each setting of the front end attenuator can cause the offset electronics to behave differently. This method of setting offset takes these items into account. The offset is an abstract value of 0 to 256 representing where the user would want the offset to be in an 8-bit full scale range. If *Update* is true, this routine sets the offset analog voltages according to these instructions and the current setting of the front end attenuator. The settings are also stored in the API. If the attenuator settings are changed, this function can be called with *Update* false to install new offset analog voltages based on offset instructions previously stored in the API and the new front end attenuator settings. This action is taken automatically by the TR122_SetChannel function.

DCAPI TR122_SaveData(BYTE Format, char *Filename);

This routine saves data for all valid samples in sample memory to the file named. Any existing file is overwritten. The validity of data is dependant on the last call to TR122_GetTriggerInformation, which should always be called before this routine is called. The format is text or binary as defined in the header file.

DCAPI TR122_SetHorizontal(DWORD nSPerDiv);

This routine is generally used only when operating the instrument in the oscilloscope mode and displaying data on the screen DIB. The sample rate is set based on the specified horizontal rate along with certain other related internal parameters.

DCAPI TR122_SetChannel(BYTE Channel, BYTE AC, BYTE Ohm50, BYTE Range, BYTE Exponent);

This routine allows direct setting of the front end attenuator for the specified channel, 1 or 2. The coupling and impedance arguments are boolean. The base full scale range is determined by *Range* as set to CHANNEL_RANGE_X control values defined in the header file. The exponent is 0, 1, or 2 for the base range of 100mV, 200mV, or 500mV multiplied by 10^0 , 10^1 , or 10^2 volts per division.

DCAPI TR122_SetupScopeMemory(void);

This routine contains only a call to TR122_SetSampleSegmentSize to set the memory geometry as expected by code related to oscilloscope mode operation.

3.2.3 Display Functions

The API library maintains for each process a device independent bitmap (DIB) which can be accessed by the display functions in the API. The DIB is created and initialized when the library is loaded. The image data appears as an oscilloscope gradicule. The size of the image is nominally 256 pixels high and 512 pixels wide. These dimensions are defined in the header file and these definitions are used for coding clarity. However, the dimensions are not intended to be changed and some code may be dependant on these dimensions, such as in the use of 256 vertical pixels for an 8-bit analog trace.

The display functions work with the DIB to include analog information, change parameters, display the image, and save the image as may be valid for various operating modes. Presented below are the exported functions with a brief description of their place in the design. They are ordered differently than in the source code for clarity of discussion.

DCAPI TR122_D_Clear(void);

The API maintains a copy of the DIB with only the background image on it. The clear function copies the background image over the final image, thereby eliminating any analog traces and other additions to the image.

DCAPI TR122_D_AssignColors(DWORD *Colors);

Colors are assigned to the elements of the DIB which include the background, gradicule, two analog traces, and the vertical trigger location bar. The colors are provided in an array pointed to by the argument and with indexes defined in the header file for elements of the DIB. The colors have the format 00BBGRR (hexadecimal) for the blue, green, and red components of each color in one 32-bit integer.

DCAPI TR122_D_TickMarks(BYTE ShowTickMarks);

This routine adds or subtracts the subgradicule tick marks from the DIB based on the argument.

DCAPI TR122_D_Placelcons(long LevelChannel1, long LevelChannel2, long LevelTrigger);

This routine places icons for the offset level for each channel and the trigger level on the DIB. The levels are given in the range 0 to 256. The use of signed arguments allows for the possibility that a caller may assign a value which is not within the vertical range of the screen (less than 0 or greater than 256). Such an icon will not be displayed for the offset levels while it will be displayed as an off screen indicator for the trigger level. The trigger level appears in the color of the trace for the current trigger source channel. If the trigger source is not a channel, then the trigger icon does not appear.

DCAPI TR122_D_SetOptions(BYTE ShowLines, BYTE ShowChannel1, BYTE ShowChannel2, BYTE ShowTrigger);

The options are elements of the DIB which should or should not be displayed as new plots are generated. These include the traces for either channel, the vertical bar at the location of the trigger, and the lines between sample points. The sample points are always displayed as part of any active trace.

DCAPI TR122_D_SetMag(BYTE Channel, BYTE VMag);

The vertical magnification in this sense identifies the 8 bits of the 12-bit sample which should be displayed. A *VMag* of 1 indicates the highest order bits. *VMag* of 2, 4, 8, or 16 amount shift the displayed quantity 1, 2, 3, or 4 bits toward the least significant bits. The argument is limited to these values and applies only to the specified channel. This will cause the plot routines to use different bits in the sample, making the display appear more sensitive by a factor of the *VMag*.

DCAPI TR122_D_ValidateTransientTimeOffset(long *PixelOffset, char *TimeAtMidScreen);

A user application will usually involve dimensioning of a display based on the pixels as they appear to the user. A plot of a valid transient record is initially displayed with the trigger at midscreen (offset of 0 samples). A transient record is plotted using one sample per horizontal pixel. This routine is a service which performs no function with the instrument or the API, but inspects the argument based on the current memory geometry and modifies the requested pixel offset if required to comply with that geometry. The corresponding offset in pixels (samples) is translated to a time based on the sample rate and written as a text string into the *TimeAtMidScreen* argument.

DCAPI TR122_D_TransientPlot(DWORD Segment, long Offset, DWORD *TraceData);

This routine plots samples of a transient record from the segment indicated to the current DIB. The offset is the number of samples before or after the trigger which should appear at the middle of the gradicule. The offset is automatically adjusted internally for the limits in the current memory geometry. Therefore, typically the caller will use the validation routine above before calling this routine. Before calling either routine the validity of a transient record in memory must be established by calling *TR122_GetTriggerInformation*. This routine also returns the samples plotted to a 512 point array referenced the *TraceData* argument. This feature provides raw sample data in an internal memory format for the benefit of applications which create plots independent of the API but need to use the API to sort out the data. This step can be safely bypassed by setting *TraceData* to NULL.

DCAPI TR122_D_SetScopeTimeOffset(long *PixelOffset, char *TimeAtMidScreen);

In the oscilloscope mode the time offset for the center of the gradicule versus the trigger location is a bit more complicated to determine because the sample information is compressed or stretched in the plot to translate the sample rate to a particular time per division. The offset is still given in terms of pixels and a string relating the time at the center line of the gradicule is returned. Because the oscilloscope mode plots new information under control of the auto thread, the time offset must be stored internally to the API for use in the plot. This function allows the higher level software to install that offset.

DCAPI TR122_D_ScopePlot(void);

The *ScopePlot* routine is generally only used by the auto thread. This export is available for primitive oscilloscope applications where the timing of data updates is controlled by the top level application. The routine automatically manipulates the DIB, collects the trigger information, compresses samples, and plots traces based on parameters installed elsewhere in the API. This routine does make the assumption that the memory geometry is as defined for the oscilloscope mode.

DCAPI TR122_D_SaveBitmap(char *Filename);

This routine saves the current DIB to a 256 color Windows bitmap file. The caller must prevent any function including the auto thread from updating the bitmap during this operation or strange results could occur.

DCAPI TR122_D_Paint(HDC hDestDC, long DestXSize, long DestYSize);

The paint routine will write the DIB to the device context specified by the handle. The size in pixels of this device context must also be specified. If the size is not the same as the DIB, the DIB will be stretched. The aspect ratio will be maintained as much as possible by excluding part of the destination from the write. However, it is possible to lose image information. It is recommended that the device context have the sample pixel dimensions as the DIB. Finally, it is important to know that the paint process will write to the foreground image of a display object. To see the new image on a screen device it may be necessary to refresh the associated object if the object automatically redisplay its background image.

3.2.4 The Auto Thread

In order to present analog data as an oscilloscope, the application must continuously collect, read, and display data. To be effective it can not be distracted from that task as the user works with the controls. There are also a number of timing functions which will work more smoothly if they are independent of anything else that is happening in the top level application. For this reason the oscilloscope mode depends on a program thread which exists entirely within the API library.

The thread is called “Auto Thread” probably mostly because of the necessity of this approach in implementing the auto trigger mode. However, the thread also automatically uses the API functions to perform transient recordings and display data to the screen DIB without intervention from the top level application. The continuously updated DIB is painted to a device context specified when the thread is started. Because the thread uses all of these functions asynchronously to the top level application, many API functions contain code which is designed to share resources with the auto thread if it is running.

Presented below are the exported functions related to the auto thread with a brief description of their place in the design. They are ordered differently than in the source code for clarity of discussion.

DCAPI TR122_AutoThread(HDC hDestDC, long XSize, long YSize, BYTE EnableThread);

This function is used to start the auto thread running or to stop it if it is running. The *EnableThread* argument specifies if the thread should be started or stopped. The other arguments are the same arguments as specified for the TR122_D_Paint function. The device context would exist in the top level application and the started instance of the auto thread would be associated with that application. It is important to know that the paint process will write to the foreground image of a display object. In the case of the auto thread the display object should not be set to automatically display its background image. The auto thread regularly updates the display regardless of the state of its data collection activities. This action in itself should be enough to satisfy the needs of the repaint action of the Windows application.

DCAPI TR122_AutoThreadTriggerMode(BYTE Mode);

The trigger mode is set with the *Mode* argument to one of the oscilloscope trigger modes as defined in the header file. This function is directly associated with the oscilloscope mode trigger options. This parameter will affect the operation of the auto thread only.

DCAPI TR122_AutoThreadTriggerRun(BYTE RunRequest);

This function directly provides the Run/Stop control functionality in oscilloscope mode. The argument will set RUN for 1 and STOP for 0. This parameter will affect the operation of the auto thread only.

4.0 Calibration

The TR122 calibration compensates for variability in the analog instrument electronics. The instrument is delivered fully calibrated and will not need any calibration efforts during normal use. Calibrations are generally good for at least one year.

There are two aspects to the calibration of the TR122. The first is a familiar process where the analog front end frequency compensation is adjusted to provide the best possible representation of the incoming signal. This calibration step involves actual adjustment of compensation capacitors in the front end electronics, similar to adjusting the compensation on an oscilloscope probe. The other aspect involves compensation for gains and offsets of the instrument electronics. This information will be discovered during calibration and stored in a non-volatile memory in the TR122 as opposed to making any physical adjustments.

The primary objective of calibration is to be sure that transient record data displayed or stored to disk matches the signal being measured within the limits of the instrument. For a transient recorder, once the front end frequency compensation is adjusted the only other items to compensate for are any irregularities in gain or offset present in the analog instrument electronics. The gain calibration information stored in the instrument will be used to adjust data retrieved from the instrument before it is presented on the screen or saved to disk. This is an automatic function of the API library, so it will be present in any application which uses this library.

In addition, the non-volatile memory is used to store information about the effect of offset voltages on the instrument electronics and what offset is present initially. This information is used to adjust the offset voltage applied such that zero offset is really zero offset and any requested offset matches the intent. This information is applied every time the offset is adjusted. The offset calibration tends to be different between settings of the instrument attenuator electronics. Therefore, the API automatically reapplies the offset when the attenuator settings are changed.

4.1 Using Calibration Features Of The TR122 Software

The TR122 software contains a complete set of calibration features. These features are generally only accessed at the factory or by metrology personnel. With these features available in the turnkey software provided, it will generally not be necessary to incorporate such features into any custom software.

To access the calibration features of the software, create a Windows shortcut or use the command line to start the software with the `-C` command line argument (i.e. `TR122.EXE -C`). This will present the window shown in Figure 4.1 with the added controls for calibration.

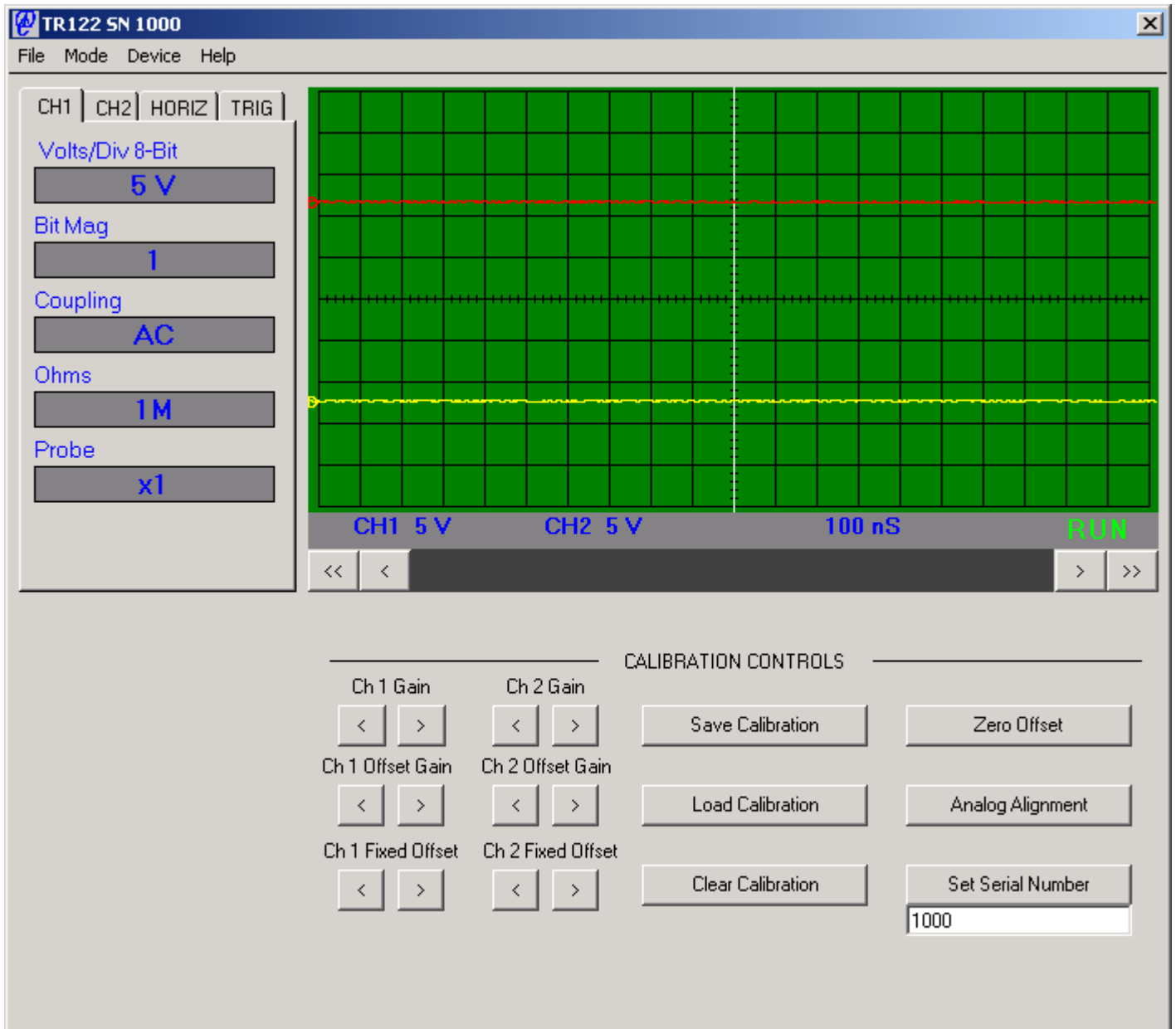


Figure 4.1 Calibration Controls

4.1.1 Analog Frequency Compensation Alignment

Generally the instrument will be fairly close to properly calibrated and the calibration procedure will be used to make minor adjustments and verify operation on a scheduled basis. The analog attenuator frequency compensation components are the most likely to drift. The status of the alignment of these components can affect the gain over frequency. Therefore, this alignment should be performed before any other step.

Adjusting the front end compensation is similar to adjusting the compensation of an oscilloscope probe. In fact, the TR122 high impedance attenuator is essentially two x10 oscilloscope probes connected in series and followed by a final attenuator and amplifier section. Figure 4.2 is representative of the design.

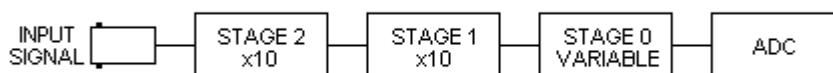


Figure 4.2 TR122 Analog Attenuator

To adjust the compensation of the attenuator components, it is necessary to identify and adjust each component independently of the others. Start by clicking on the *Zero Offset* button to return the traces to midscreen. It may be helpful to display only one trace at a time. All of these adjustments should be done in oscilloscope mode and are best done with a horizontal setting and input frequency which provides the opportunity to observe step transitions over relatively long periods of time. Oscilloscope frequency compensation is typically adjusted at 1 KHz, and the transient recorder will not be much different except that a higher frequency might be selected for display convenience. The display must be sufficient to observe affects on step transitions over time and thus over a range of frequencies. It will become apparent that the ability to apply test waveforms of various frequencies is also helpful in checking results.

Set the vertical controls for DC coupling. Apply a 100 mV to 200 mV peak to peak square wave to the channel being adjusted. The amplitude is not important, but it should be certain that the waveform has as square an edge as possible. Click on the *Analog Alignment* button. The attenuator will adjust so that only the stage 0 attenuator is affecting the signal and a message will display. Leave the message on the screen. Use a plastic alignment tool to adjust the compensation capacitor furthest from the front of the unit. The alignment capacitors appear through holes in the attenuator module printed wiring board behind each channel input connector and offset towards the analog-to-digital converter integrated circuit (chip) for each channel. They appear in the order shown in Figure 4.2 from the front panel connection side of the board and towards the converter chip. Adjust the alignment for stage 0 until the signal displayed on the oscilloscope screen is as accurate a representation of the incoming signal as possible with respect to the shape of the rising and falling edges. Due to the limited bandwidth of the instrument, the signal will never be square, but it will be close, particularly at low frequencies.

Note: Adjustment of the attenuator compensation capacitors requires a special tuning tool. This tool is available from Data Design. Use of any other tool for this purpose will be difficult and could damage the component.

With the stage 0 attenuator compensation aligned, the front end x10 attenuators can be aligned. To separate the action of each and because the available test signals are likely to be limited to a few volts, these attenuators will have to be adjusted one at a time. This is a departure from normal instrument operation. Click the *Ok* button to clear the message for stage 0. A similar message will appear for stage 1. Apply a signal with an amplitude in the range of a few volts and use the same method as above to adjust the compensation trimmer for stage 1. Click *Ok* again to repeat for stage 2 with the same input signal. Click *Ok* a final time and the instrument will return to its original settings.

This completes the attenuator compensation calibration. Repeat this procedure for each channel until the performance is satisfactory. It will be helpful to examine performance on rising and falling edges over a range of input signal frequencies and instrument settings

4.1.2 Stored Calibration Controls

Other than analog compensation, all calibration information is stored in non-volatile memory in the instrument hardware. This information is retrieved and applied by the software when the application is started. This operation occurs at the level of the API library so it will be transparent to any top level application which uses the API.

As calibration items are adjusted, they are adjusted only within the API and not saved back to the instrument. The button *Load Calibration* is available to load the information from the instrument as it was before calibration began. The button *Save Calibration* will save the calibration information in the API, as adjusted by any updates, to the instrument non-volatile memory. The button *Clear Calibration* will return all gains and offsets to nominal values and clear the non-volatile memory on the instrument. This step is rather drastic, and calibration can usually be completed without the need to do this. Calibration will be a quicker process if it is not cleared, because the user can take advantage of previous calibration settings which are likely close to where they need to be. If significant repairs have been made to a damaged instrument it may help to clear the calibration memory.

The serial number of the instrument is also stored in non-volatile memory. The button *Set Serial Number* allows this number to be changed to the displayed value in the edit box. The serial number on the product labeling will always be the official serial number of the unit for purposes of communicating with Data Design regarding the module. The number stored in the instrument electronics allows the user to easily identify the unit in use. For this reason it is recommended that this number never be changed. But then in the spirit of open source, the user is provided with the ability to do bad things as well as good things.

4.1.3 Adjusting The Calibration Data

Several controls < and > are provided to adjust each of three calibration points for each of the channels. These calibration points must be adjusted for each of the nine settings of the input attenuator. In total then there are $2 \times 9 \times 3 = 54$ settings to be made. To be sure that everything is adjusted properly it helps to be systematic about it. The following is a suggested approach.

Set the front end to DC coupling and 50 ohms and be sure that no signal is applied. The *Bit Mag* setting must be set to 1. Set the display to show one channel at a time and complete all of these steps for that channel before moving on to the other channel. For the channel being calibrated, start with the 100 mV per division setting and work upwards. For each setting proceed as follows.

- 1) Click *Zero Offset* to bring the trace to the middle of the gradicule. If properly adjusted, the trace should appear exactly in the middle. If it does not, use the *Fixed Offset* controls for that channel to move the trace to the middle of the gradicule.
- 2) Move the offset towards the top of the screen. If properly adjusted, the trace should be at the sample location as the offset arrow. If it is not, use the *Offset Gain* control to bring the trace into agreement with the icon. Move the offset towards the bottom of the screen and repeat. It may take a few tries to reach a good compromise on offset gain over all locations on the screen. It will never be perfect at all locations, but should be quite close.

Return the attenuator to the 100 mV setting and apply an appropriate amplitude signal to the channel under test from a calibrated source. It may be helpful to apply a signal which covers an integer number of divisions and preferably covers half or more of the full scale range. At higher ranges, this may not always be possible. A sine wave is generally the best choice for examining amplitude. For each setting of the attenuator proceed as follows, adjusting the amplitude of the known signal as needed.

- 1) Adjust the *Gain* control so that the amplitude displayed on the gradicule matches the amplitude of the known signal.
- 2) Change the frequency of the incoming signal to an order of magnitude higher or lower than the current setting without changing its amplitude. Change the *Sec/Div* control to view and verify the signal amplitude across at least two frequencies well within the instrument bandwidth. Good choices for calibration signals are 1 MHz and 10 MHz. This approach will allow multiple viewings of the same information on different settings of the instrument, making it more likely to converge on a good calibration setting.

When all calibrations have been made, use the *Save Calibration* button to write the new data to the instrument non-volatile memory. To protect tedious calibration work from interruption and data loss, it may be advisable to perform this step at several points during the process. Success of the calibration process should be reviewed at all settings after the process is complete. This concludes instrument calibration.

TR122 and *TR122 Software* are products of Data Design Corporation. No portion of this manual or any accompanying software may be reproduced by any means for any purpose other than for user archives and application of the *TR122* products without the express written consent of the copyright owner.

Copyright © 2004 Data Design Corporation
All rights reserved.

PRODUCT WARRANTY:

1. Product. The "Product" referred to herein is the instrumentation product *TR122* and the software product *TR122 Software*. The product as defined does not include the *TR122 Software Source Code* or documentation of such source code.

2. Limited Warranty. Data Design Corporation warrants that the Product will perform substantially in accordance with the accompanying documentation for a period of one year from the date of receipt. Any implied warranties on the Product are limited to one year. SOME STATES DO NOT ALLOW LIMITATIONS ON DURATION OF AN IMPLIED WARRANTY, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

3. Remedies. The entire liability of Data Design Corporations and/or its suppliers and your exclusive remedy shall be, at the option of Data Design Corporation, either (a) return of the purchase price you paid or (b) repair or replacement of the Product that does not meet the Limited Warranty and that is returned to the place of purchase with a copy of your receipt. The Limited Warranty is void if failure of the Product has resulted from accident, abuse, or misapplication. Any replacement Product will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. THE FOREGOING CONSTITUTES YOUR SOLE AND EXCLUSIVE REMEDY UNDER THIS WARRANTY. EXCEPT FOR THE WARRANTIES SET FORTH ABOVE, THE PRODUCT AND DOCUMENTATION ARE PROVIDED "AS IS," AND DATA DESIGN CORPORATION AND/OR ITS SUPPLIERS DISCLAIM ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

4. Limitations of Liability. The cumulative liability of Data Design Corporation and/or its suppliers to you or any other party for any loss or damages resulting from any claims, demands, or actions arising out of or relating to this warranty shall not exceed the purchase price you paid for the Product and documentation. In no event shall Data Design Corporation and/or its suppliers be liable for any indirect, incidental, consequential, special, or exemplary damages, loss of business profits, business interruption, or loss of business information, even if advised of the possibility of such damages. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

TRADEMARKS:

Commercial names and trademarks are used for identification purposes only and are recognized as the property of the respective trademark owners.

REVISION LEVEL:

Firmware Date: December 20, 2004
Software Version: 1.20
Manual Release Date: April 14, 2005

TERMS AND CONDITIONS OF LICENSE REGARDING *TR122 SOFTWARE SOURCE CODE*:

PLEASE REVIEW THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE DOWNLOADING OR USING ANY SOURCE CODE OR SOURCE CODE DOCUMENTATION RELATED TO THE TR122 PRODUCT (the "SOURCE CODE"). BY USING THE SOURCE CODE, YOU INDICATE YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS, WHICH CONSTITUTE A LICENSE AGREEMENT (the "AGREEMENT") BETWEEN YOU AND DATA DESIGN CORPORATION ("DATA DESIGN"). IN THE EVENT THAT YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT DOWNLOAD, COPY, OR USE THE SOURCE CODE. THE SOURCE CODE IS GOVERNED SOLELY BY THIS AGREEMENT AND NOT BY ANY OTHER AGREEMENT OR LICENSE THAT YOU MAY HAVE WITH DATA DESIGN, UNLESS PROVIDED IN WRITING. BY DOWNLOADING OR USING THE SOURCE CODE, LICENSEE ACKNOWLEDGES THAT HE/SHE HAS READ THIS AGREEMENT, UNDERSTANDS IT, AND AGREES TO BE BOUND BY ITS TERMS AND CONDITIONS

Under this Agreement, Data Design is providing source code for software related to the TR122 transient recorder product as delivered and as may be amended or augmented via various mediums from time to time. LICENSEE MAY NOT USE, COPY, MODIFY, DISTRIBUTE, SUBLICENSE OR TRANSFER THE SOURCE CODE, OR MERGED OR COMBINED PORTION THEREOF, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS AGREEMENT. THE SOURCE CODE MAY NOT BE USED FOR ANY PURPOSE OTHER THAN THE DEVELOPMENT OF SOFTWARE APPLICATIONS FOR THE TR122 PRODUCT.

"Source Code" means one or more hardware or software design files in source format.

"Licensee" means YOU, the user of the Source Code, under this Agreement.

License Subject to the terms and conditions of this Agreement, Data Design grants to Licensee, a non-transferable, non-exclusive, perpetual license to use the Source Code. The Source Code, and the algorithms, concepts, techniques, methods and processes embodied therein, are proprietary to Data Design. Data Design retains all rights with respect to the Source Code, including any copyright, patent, and other proprietary rights, not expressly granted herein.

Licensee may implement the Source Code in a larger work, modify the Source Code, and create derivative works thereof. However, no copy or derivative work may be presented for sale in any venue without the express written consent of Data Design. All such works must be applicable only to use of the TR122 product. Any copy or portion of the Source Code, including any derivative works thereof, including any portion merged into a design and any design or product that incorporates any portion of the Source Code, will continue to be subject to the terms and conditions of this Agreement.

Any direct copies of the Source Code made by Licensee under this Agreement shall include all intellectual property notices, including copyright and proprietary rights notices, appearing on the Source Code as provided by Data Design.

No License To Trademarks Data Design reserves exclusive use of any and all of its trademarks.

Term This Agreement is effective until terminated. Licensee may terminate this agreement at any time by destroying the Source Code together with all copies, derivative works and portions thereof in any form (including any portions merged into a design). This Agreement will also terminate immediately upon Licensee's material breach or if Licensee fails to comply with any term or condition of this Agreement. Upon any termination of this Agreement, the license and rights of Licensee under this Agreement shall terminate, and Licensee shall destroy the Source Code, including all copies, derivative works and portions thereof in any form (including any portions thereof merged into a design).

Payment The Source Code is being provided to Licensee at no cost, except to the extent that the Source Code may only be used in conjunction with a product which Data Design offers for sale. Data Design shall be compensated for any customization of the Source Code performed by Data Design as agreed upon in writing by the parties.

No Warranties, Support, or Maintenance THE SOURCE CODE IS PROVIDED TO LICENSEE "AS-IS". LICENSEE ALSO AGREES THAT DATA DESIGN DOES NOT PROVIDE MAINTENANCE OR SUPPORT FOR THE SOURCE CODE. NO WARRANTIES, REPRESENTATIONS, OR GUARANTIES, EITHER EXPRESS OR IMPLIED, ARE MADE WITH RESPECT TO THE SOURCE CODE, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT. LICENSEE ASSUMES THE ENTIRE RISK AS TO THE QUALITY, COMPLETENESS AND PERFORMANCE OF THE SOURCE CODE AND ANY DESIGN OR PRODUCT IN WHICH THE SOURCE CODE MAY BE USED. SHOULD THE SOURCE CODE PROVE DEFECTIVE, DATA DESIGN ASSUMES NO LIABILITY FOR ANY COST OF ANY NECESSARY SERVICING, REPAIR, OR CORRECTION. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you in full, but shall be interpreted to apply to the maximum extent permissible under applicable law. DATA DESIGN DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOURCE CODE WILL MEET ANY REQUIREMENTS, OR THAT THE OPERATION OF THE SOURCE CODE WILL BE UNINTERRUPTED OR ERROR-FREE. LICENSEE ALSO ASSUMES RESPONSIBILITY FOR THE SELECTION OF THE SOURCE CODE TO ACHIEVE INTENDED RESULTS AND FOR THE INSTALLATION, USE, AND RESULTS OBTAINED FROM THE SOURCE CODE.

Limitations of Liability In no event shall the aggregate liability of Data Design relating to this Agreement or the subject matter hereof under any legal theory (whether in tort, contract, or otherwise), including any liability for any loss or damages directly or indirectly suffered by Licensee relating to the Source Code, exceed One Dollar (\$1.00). IN NO EVENT SHALL DATA DESIGN BE LIABLE UNDER ANY LEGAL THEORY, WHETHER IN TORT, CONTRACT OR OTHERWISE (a) FOR ANY LOST PROFITS, LOST REVENUE OR LOST BUSINESS, (b) FOR ANY LOSS OF OR DAMAGES TO OTHER SOFTWARE OR DATA, OR (c) FOR ANY INCIDENTAL, INDIRECT, CONSEQUENTIAL, PUNITIVE OR SPECIAL DAMAGES RELATING TO THIS AGREEMENT OR THE SUBJECT MATTER HEREOF, INCLUDING BUT NOT LIMITED TO THE DELIVERY, USE, SUPPORT, OPERATION OR FAILURE OF THE SOURCE CODE, EVEN IF DATA DESIGN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LIABILITY AND NOTWITHSTANDING ANY FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDY STATED HEREIN. LICENSEE ACKNOWLEDGES THAT DATA DESIGN HAS NO RESPONSIBILITY OR DUTY TO DEFEND, INDEMNIFY OR HOLD LICENSEE HARMLESS FROM AND AGAINST ANY CLAIMS, SUITS, PROCEEDINGS, DAMAGES, LOSSES, COSTS AND EXPENSES, BASED ON PATENT OR OTHER INTELLECTUAL PROPERTY CLAIMS.

U.S. Government Restricted Rights If Licensee is an agency or instrumentality of the United States Government, the Source Code and related documentation are "commercial computer software" and "commercial computer software documentation", and pursuant to FAR 12.212 or DFARS 227.7202, and their successors, as applicable, use, reproduction and disclosure of the Source Code and related documentation are governed by the terms of this Agreement.

Severability If any provision of this agreement is held by a court of competent jurisdiction to be legally ineffective or unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable and the validity of the remaining provisions shall not be affected.